Shell 常用命令

目录

Shell 基本命令

> 查看基础命令

shell 命令行模式下输入 `help` 查看手册基础命令；

- { **【command】 --help** } 查看命令帮助
- { **info 【command】** } 查看命令文档

bash 命令行模式下输入 `bash` 然后再输入 `help` 查看手册基础命令；

- **man bash** 查看bash的命令帮助
- **info bash** 查看bash的文档

我们需要展示一个命令行的小技巧：可以使用分号作为分隔符，在命令行中一次性输入多个命令。就像下面这样：

command1; command2; command3...

基础命令

`date` 查看当前时间日期
`cal` 查看当前月份日历
`df` 当前磁盘可用空间
`free` 当前可用内存容量

操作文件和目录

`cd 目录` 进入目录
`pwd` 输出当前工作目录
`ls` 查看目录
`file` 确定文件类型
`less` 查看文件内容
`mkdir` 创建目录
`mv` 移动文件
`cp` 复制文件
`ln` 创建硬链接和符号链接
`rm` 删除文件

命令类型

`type` 显示命令类型
`which` 显示可执行文件位置
`help` 获取Shell内建命令的帮助信息
`man` **显示命令的手册页**
`apropos` 显示适合的命令清单
`whatis` 显示手册页的简述
`info` 显示命令的info条目
`alias` 创建自己的命令

“I/O”重定向 input/output

`cat` 拼接文件
`sort` 排序文本行
`uniq` 报告或忽略重复的行
`wc` 统计文件中换行符，单词以及字节的数量
`grep` 输出与模式匹配的行
`head` 输出文件的开头部分
`tail` 输出文件的结尾部分
`tee` 读取标准输入并将输出结果写入标准输出的文件 **ls /usr/bin | tee ls.txt | grep zi**
p

Shell查看

`echo` 显示一行文本 **echo {1..15}; echo ***

键盘技巧

`clear` 清除终端屏幕

`history` 显示或操作命令历史列表

权限

`id` 查看用户的身份

`chmod` 修改文件模式

`umask` 设置默认权限

`su` 以其他用户身份启动Shell

`sudo` 以其他用户身份执行命令

`chown` 更改文件属主和属组

`chgrp` 更改文件属组

`passwd` 修改密码

进程

`ps` 查看进程

`top` 动态查看进程

`jobs` 查看启动的作业

`fg` 将作业置于前台

`bg` 将作业置于后台

`kill` 向进程发送信号

`killall` 按名称终止进程

`shutdown` 关闭或重启系统

关闭系统

`halt`

`poweroff`

`reboot`

`shutdown`

环境

`printenv` 显示部分或全部环境变量

`set` 显示Shell变量和环境变量

`echo` 查看变量内容

`alias` 查看命令别名

存储介质

`mount` 挂载文件系统

`umount` 卸载文件系统

`fsck` 检查和修复文件系统

`fdisk` 操作分区

`mkfs` 创建新的文件系统

`dd` 转换和复制文件

`genisoimage (mkisofs)` 创建ISO 9660映像文件

`wodim` 擦除和刻录光学存储介质

`md5sum` 计算MD5校验和

联网

`ping` ： 向网络主机发送ICMP ECHO_REQUEST分组。

`traceroute` ： 输出分组抵达网络主机所经历的路线。

`ip` ： 显示/操作路由、设备、策略路由、隧道。

`netstat` ： 输出网络连接、路由表、接口统计、伪装连接、多播成员关系。

`ftp` ： 互联网文件传输程序。

`wget` ： 非交互式网络下载工具。

`ssh` ： OpenSSH客户端（远程登录程序）。

`scp` 和 `sftp` ： 在网络上复制文件。

查找文件

`locate` ： 按照路径名查找文件。

`find` ： 在目录中查找文件。

我们还会介绍一个经常配合文件查找命令来处理结果文件的命令。

`xargs` ： 通过标准输入构建并执行命令。

除此之外，还有两个能够派上用场的命令。

`touch` ： 修改文件时间。

`stat` ： 显示文件或文件系统状态。

文本处理

`cat` ： 拼接文件。

`sort` ： 排序文本行。

`uniq` ： 报告或忽略重复的行。

`cut` ： 从每行中删除部分内容。

`paste` ： 合并行。

`join` ： 连接两个文件中具有公共字段的行。

`comm` ： 逐行比较两个已排序的文件。

`diff` ： 逐行比较文件。

`patch` ： 对原文件应用diff文件。

`tr` ： 转写或删除字符。

`sed` ： 用于文本过滤和转换的流编辑器。

`aspell` ： 交互式拼写检查器。

格式化输出

`nl` ： 对行进行编号。

`fold` ： 在指定长度处折行。

`fmt` ： 一个简单的文本格式化工具。

`pr` ： 格式化要输出的文本。

`printf` ： 格式化并输出数据。

`groff` ： 文档格式化系统。

打印

`pr` ： 转换要打印的文本文件。

`lpr` ： 以Berkeley风格打印文件。

`lp` ： 以System V风格打印文件。

`a2ps` ： 在PostScript打印机上打印文件。

`lpstat` ： 显示打印系统状态信息。

`lpq` ： 显示打印队列状态。

`lprm` / `cancel` ： 取消打印作业。

编译程序

`make` ： 程序维护工具。

脚本

文本

```
#!/bin/bash

# 程序输出一个系统信息页

TITLE="System Information Report For $HOSTNAME"
CURRENT_TIME="$(date +"%x %r %Z")"
TIMESTAMP="Generated $CURRENT_TIME, by $USER"

cat << _EOF_
<html>
        <head>
                <title>$TITLE</title>
        </head>
        <body>
                <h1>$TITLE</h1>
                <p>$TIMESTAMP</p>
                $(report_uptime)
                $(report_disk_space)
                $(report_home_space)
        </body>
</html>
_EOF_
```

if分支

```
report_home_space () {
      if [[ "$(id -u)" -eq 0 ]]; then
            cat <<- _EOF_
                  <h2>Home Space Utilization (All Users)</h2>
                  <pre>$(du -sh /home/*)</pre>
                  _EOF_
      else
            cat <<- _EOF_
                  <h2>Home Space Utilization ($USER)</h2>
                  <pre>$(du -sh $HOME)</pre>
                  _EOF_
      fi
      return
}
```

读取键盘输入

```
#!/bin/bash

# read-menu: 菜单驱动的系统信息程序

clear
echo "
Please Select:

1. Display System Information
2. Display Disk Space
3. Display Home Space Utilization
0. Quit
"
read -p "Enter selection [0-3] > "

if [[ "$REPLY" =~ ^[0-3]$ ]]; then
      if [[ "$REPLY" == 0 ]]; then
            echo "Program terminated."
            exit
      fi
      if [[ "$REPLY" == 1 ]]; then
            echo "Hostname: $HOSTNAME"
            uptime
            exit
      fi
      if [[ "$REPLY" == 2 ]]; then
            df -h
            exit
      fi
      if [[ "$REPLY" == 3 ]]; then
            if [[ "$(id -u)" -eq 0 ]]; then
                  echo "Home Space Utilization (All Users)"
                  du -sh /home/*
            else
                  echo "Home Space Utilization ($USER)"
                  du -sh "$HOME"
            fi
            exit
      fi
else
      echo "Invalid entry." >&2
      exit 1
fi
```

流程控制：while/until循环

until

```
#!/bin/bash

# until-count: 显示一系列数字

count=1
until [[ "$count" -gt 5 ]]; do
      echo "$count"
      count=$((count + 1))
done
echo "Finished."
```

while

```
#!/bin/bash

# while-read2: 从文件读取行

sort -k 1,1 -k 2n distros.txt | while read distro version release; do
      printf "Distro: %s\tVersion: %s\tReleased: %s\n"\
            "$distro"\
            "$version"\
            "$release"
done
```

故障诊断

```
#!/bin/bash

# 错误：演示常见错误的脚本
number=1
if [ $number = 1 ]; then
      echo "Number is equal to 1."
else
      echo "Number is not equal to 1."
fi
```

以下缺少引号测试报错：

```
#!/bin/bash

# 错误：演示常见错误的脚本

number=1

if [ $number = 1 ]; then
      echo "Number is equal to 1.
else
      echo "Number is not equal to 1."
fi
```

在这个简单的例子中，我们只显示了变量number的值，并使用注释标记出了额外添加的行，以便于后续的识别和删除。这项技术在观察脚本中的循环和算术运算行为时尤其有用。

```
#!/bin/bash

# 错误：演示常见错误的脚本

number=1

echo "number=$number" # 调试
set -x # 打开跟踪
if [ $number = 1 ]; then
      echo "Number is equal to 1."
else
      echo "Number is not equal to 1."
fi
set +x # 关闭跟踪
```

case分支

```
#!/bin/bash

# case4-2: 测试一个字符

read -n 1 -p "Type a character > "
echo
case "$REPLY" in
     [[:upper:]])   echo "'$REPLY' is upper case." ;;&
     [[:lower:]])   echo "'$REPLY' is lower case." ;;&
     [[:alpha:]])   echo "'$REPLY' is alphabetic." ;;&
     [[:digit:]])   echo "'$REPLY' is a digit." ;;&
     [[:graph:]])   echo "'$REPLY' is a visible character." ;;&
     [[:punct:]])   echo "'$REPLY' is a punctuation symbol." ;;&
     [[:space:]])   echo "'$REPLY' is a whitespace character." ;;&
     [[:xdigit:]])  echo "'$REPLY' is a hexadecimal digit." ;;&
esac
```

执行结果如下：

```
[me@linuxbox ~]$ case4-2
Type a character > a
'a' is lower case.
'a' is alphabetic.
'a' is a visible character.
'a' is a hexadecimal digit.
```

位置参数

```
#!/bin/bash

# sys_info_page: 程序输出系统显示页

PROGNAME="$(basename "$0")"
TITLE="System Information Report For $HOSTNAME"
CURRENT_TIME="$(date +"%x %r %Z")"
TIMESTAMP="Generated $CURRENT_TIME, by $USER"

report_uptime () {
      cat <<- _EOF_
            <h2>System Uptime</h2>
            <pre>$(uptime)</pre>
            _EOF_
      return
}

report_disk_space () {
      cat <<- _EOF_
            <h2>Disk Space Utilization</h2>
            <pre>$(df -h)</PRE>
            _EOF_
      return
}

report_home_space () {
      if [[ "$(id -u)" -eq 0 ]]; then
            cat <<- _EOF_
                  <h2>Home Space Utilization (All Users)</h2>
                  <pre>$(du -sh /home/*)</pre>
                  _EOF_
      else
            cat <<- _EOF_
                  <h2>Home Space Utilization ($USER)</h2>
                  <pre>$(du -sh "$HOME")</pre>
                  _EOF_
      fi
      return
}

usage () {
      echo "$PROGNAME: usage: $PROGNAME [-f file | -i]"
      return
}

write_html_page () {
      cat <<- _EOF_
      <html>
            <head>
                  <title>$TITLE</title>
            </head>
            <body>
                  <h1>$TITLE</h1>
                  <p>$TIMESTAMP</p>
                  $(report_uptime)
                  $(report_disk_space)
                  $(report_home_space)
            </body>
      </html>
      _EOF_
      return
}

# 进程命令行选项

interactive=
filename=

while [[ -n "$1" ]]; do
    case "$1" in
        -f | --file)        shift
                            filename="$1"
                            ;;
        -i | --interactive) interactive=1
                            ;;
        -h | --help)        usage
                            exit
                            ;;
        *)                  usage >&2
                            exit 1
                            ;;
    esac
    shift
```

```
done

# 交互模式

if [[ -n "$interactive" ]]; then
    while true; do
        read -p "Enter name of output file: " filename
        if [[ -e "$filename" ]]; then
            read -p "'$filename' exists. Overwrite? [y/n/q] > "
            case "$REPLY" in
                Y|y)    break
                        ;;
                Q|q)    echo "Program terminated."
                        exit
                        ;;
                *)      continue
                        ;;
            esac
        elif [[ -z "$filename" ]]; then
            continue
        else
            break
        fi
    done
fi

# 输出HTML页

if [[ -n "$filename" ]]; then
      if touch "$filename" && [[ -f "$filename" ]]; then
            write_html_page > "$filename"
      else
            echo "$PROGNAME: Cannot write file '$filename'" >&2
            exit 1
      fi
else
      write_html_page
fi
```

流程控制 for循环

```
report_home_space () {

      local format="%8s%10s%10s\n"
      local i dir_list total_files total_dirs total_size user_name

      if [[ "$(id -u)" -eq 0 ]]; then
            dir_list=/home/*
            user_name="All Users"
      else
            dir_list="$HOME"
            user_name="$USER"
      fi

      echo "<h2>Home Space Utilization ($user_name)</h2>"

      for i in $dir_list; do

            total_files="$(find "$i" -type f | wc -l)"
            total_dirs="$(find "$i" -type d | wc -l)"
            total_size="$(du -sh "$i" | cut -f 1)"

            echo "<h3>$i</h3>"
            echo "<pre>"
            printf "$format" "Dirs" "Files" "Size"
            printf "$format" "----" "-----" "----"
            printf "$format" "$total_dirs" "$total_files" "$total_size
"
            echo "</pre>"
      done
      return
}
```

字符串和数字

```
#!/bin/bash

# loan-calc: 计算按月偿还的贷款

PROGNAME="${0##*/}" # 使用参数扩展来获取返回路径的文件名

usage () {
      cat <<- EOF
      Usage: $PROGNAME PRINCIPAL INTEREST MONTHS

      Where:

      PRINCIPAL is the amount of the loan.
      INTEREST is the APR as a number (7% = 0.07).
      MONTHS is the length of the loan's term.

      EOF
}

if (($# != 3)); then
      usage
      exit 1
fi

principal=$1
interest=$2
months=$3

bc <<- EOF
      scale = 10
      i = $interest/12
      p = $principal
      n = $months
      a = p * ((i * ((1 + i) ^ n))/(((1 + i) ^ n) - 1))
      print a, "\n"
EOF
```

数据

```
declare -A colors
colors["red"]="#ff0000"
colors["green"]="#00ff00"
colors["blue"]="#0000ff"
```

访问：

```
echo ${colors["blue"]}
```

其它命令

```
#!/bin/bash

# async-parent: 异步执行演示（父脚本）

echo "Parent: starting..."

echo "Parent: launching child script..."
async-child &
pid=$!
echo "Parent: child (PID= $pid) launched."

echo "Parent: continuing..."
sleep 2

echo "Parent: pausing to wait for child to finish..."
wait "$pid"

echo "Parent: child is finished. Continuing..."
echo "Parent: parent is done. Exiting."
```

结果：

```
[me@linuxbox ~]$ async-parent
Parent: starting...
Parent: launching child script...
Parent: child (PID= 6741) launched.
Parent: continuing...
Child: child is running...
Parent: pausing to wait for child to finish...
Child: child is done. Exiting.
Parent: child is finished. Continuing...
Parent: parent is done. Exiting.
```

....

posted @ 2021-08-22 23:44 徐锅 阅读(1722) 评论(0) 编辑 收藏 举报